PowerShell and Object-Level Math Functions

By: Tim Smith | Read Comments | Related Tips: More > PowerShell

Problem

We use PowerShell scripts to extract from APIs and we often want to convert our numerical values before inputting them into a table, such as rounding some decimals, validating ranges, etc. How can we do this in PowerShell?

Solution

Microsoft Windows offers a mathematics class (Math) that comes with some methods for manipulating numbers. When I'm dealing with sets of data, I prefer to use T-SQL, while with data objects themselves I prefer to use an object-oriented language, like PowerShell or C#. A simple example would be rounding an extracted data point from an API (specific point), versus taking the average of one billion rows of data (set). The languages are optimized for this purpose, so if I need to manipulate object-level data pre-import, it makes more sense to keep that in the extract layer. In addition, manipulating data before an import can also be useful when applying the definition of the table that will receive the data.

In this tip, we'll look at some functionality that we can create using Microsoft's Math class.

Round in PowerShell

Let's look at a simple example of using the Math class and rounding to the nearest integer and rounding to the second decimal place:

```
$round = 12.346
[Math]::Round($round)
[Math]::Round($round,2)
<#
### Output:
12
12.35
#>
```

Also, just a quick elementary reminder that a number which will be round up following a 9 will return a higher value for the number before the 9:

```
$skip = 23.4967
[Math]::Round($skip,2)
[Math]::Round($skip,3)
<#
### Output:
23.5
```

```
23.497
#>
```

In some cases, we may want to apply a math operation to a value before we round the returned value and we can do this with the operation first, then round the result - imagine taking a daily value multiplied by 7 to return the weekly value. In this example, we do this following the order of operations within the first parameter of the round method:

```
$daily = 0.462
[Math]::Round(($daily*7),2)
<#
### Output:
3.23
#>
```

Floor and Ceiling in PowerShell

The math class also includes methods that allow us to return the highest and lowest integer range of a decimal; for an example if we use the *$skip* example above this, we would get 23 on the floor and 24 on the ceiling:

```
$skip = 23.4967
[Math]::Floor($skip)
[Math]::Ceiling($skip)
<#
### Output:
23
24
#>
```

If we're importing data on an object level basis, it may be faster to transform the data on this layer. Keep in mind that T-SQL offers both floor and ceiling functions as well and when we're dealing with large sets of data, T-SQL can apply this quickly on these sets.

Trigonometry in PowerShell

The Math class comes with two properties as well, pi and the natural logarithm (e). Pi becomes helpful when using the trigonometry methods in the Math class, as we must first get the angle in radians to obtain the trigonometric values; for an example suppose that we have an adjacent length of 1000 meters sharing an angle with the hypotenuse of a triangle of 30 degrees and we want to know the length of the opposite side (measuring distance is a very common application of trigonometry):

```
### We first want to get the tan of 30 degrees in radians:
(30*([Math]::PI/180))
### We can wrap the above, which returns the radian value, in the method tan
[Math]::Tan((30*([Math]::PI/180)))
### Finally, what is the opposite side's length:
1000*([Math]::Tan((30*([Math]::PI/180))))
<#
### Output:
577.3502
#>
### And if we want to round it to the second decimal place:
[Math]::Round(1000*([Math]::Tan((30*([Math]::PI/180)))),2)
```

```
<#
### Output:
577.35
#>
```

The above example shows how we can apply the math class in many places, using both its methods and properties - if we wanted to repeat some of it - for example, keep radians in an object - we could use the below and get the same result:

```
[decimal]$radians = ([Math]::PI/180)
[Math]::Round(1000*([Math]::Tan((30*$radians))),2)
[Math]::Round(1000*([Math]::Tan((40*$radians))),2)
<#
### Output:
577.35
839.1
#>
```

Here we save radians as an object and re-use this object to get the tan of both 30 and 40 degrees, and return the opposite side of an adjacent side of 1000 meters.

Of course, these are not the only methods in the math class - this tip shows what we can do an object level, if we receive data like this before saving to a database. You can see other methods available, if there may be other measures you need and most math functions that are not methods can be derived - like average is the sum of a set divided by the count of the set (consider that average, like median, may involve a large set where T-SQL would perform better).

Next Steps

- You can apply these methods within the functions, before the functions, or independent of functions.
- For sets and aggregates, T-SQL offers an alternative when performance is greater on sets.
- Review the math class to view other tools available to you.

Last Update: 2017-03-28

About the author

Tim Smith works as a DBA and developer and also teaches Automating ETL on Udemy.

View all my tips

Related Resources

- More SQL Server DBA Tips...